UnusedWords

Part of the content of this page is outdated.
Why some words are marked as unused when they in fact are not

Technical description
The *get_strings.php* script collects all strings that are enclosed by the **tra** function in .php files and all strings that are enclosed by **{tr}**...**{/tr}** in .tpl files. The problem is that some tiki functions/features generates strings on the fly (reportedly this happens for flags and possibly in other places), which means that these strings can not be found and will not end up in .../tiki/lang//language.php.

[illegible]
sylvie: The strings must be added in the .php or .tpl into a special comment to be recognized by get_strings
//get_strings tra("the_string") in a php file
{*get_strings {tr}the_string{/tr} *} in a tpl file
Note there is no space before get_strings
So the developpers or the the translators who find such a string must add this kind of comments to have get_strings collecting automatically the string
See an example in tikilib.php
$action = "Removed"; //get_strings tra("Removed");

Bad solution (before Tikiwiki 1.9)
Some translators have "solved" this by adding these strings in language.php. The problem with this aproach is that subsequent usage of get_strings.php will mark these strings as unused. This is a problem when *get_strings.php* is used on an old version of language.php but a new version of Tiki. In that case legitimately unused strings will be mixed with strings that some translators have added manually. These strings will also only be visible for translators of that language.

Suggested short and long term solutions
It is of course possible to extend get_strings.php to find out which the dynamically generated strings are (preferably through plugin modules - contact me for suggestions on plugin architecture either through commenting this article, or through UserPagedocekal). This should be considered a long term solution.

In the mean time the following method is used (**since Tikiwiki 1.9RC3**) to export dynamically generated strings to *get_strings.php*.

You can make a *.php* file (either one global or one for each type of dynamically created strings) as the example below.

```
<?php /* * The listing associates country names used as filenames for flags in Tikiwiki for language translation */ // This script is only for language translations - so its better to die if called directly. if (strpos($_SERVER["SCRIPT_NAME"],basename(__FILE__)) !== false) { header("location: index.php"); exit; } // Here come the dynamically generated strings for img/flags/*.gif tra('American_Samoa'); tra('Angola'); tra('Antigua'); tra('Argentina'); tra('Armenia'); tra('Australia'); tra('Austria'); tra('Bahamas'); // etc. ?>
```

Adding such a file to a suitable place in the tiki file structure achieves two ends:

1. The strings no longer appear in the untranslated words section
2. Other translators are aware of that these strings should be translated to and they do not have to rely on trail and error to find out which strings should be translated/added to language.php

Source of the above code snippet

{CVS()}img/flags/flagnames.php{CVS}

Note: this one requires an exception (hardwiring) in get_strings.php since img dir is excluded from scanning of strings by default

---

Chealer9 comments : That looks like a good solution...could you mention a default file to do that.

---

docekal: My suggestion to this temporary solution is to do it like sylvie suggested. Put the strings in the file where they are generated just enclose them in an **if (false)** e.g.:
if (false) {
tra ('flagname1');
tra ('flagname2');
tra ('flagname3');
}
sylvie: The important point is to keep the constant and the translation call close to each other. My favorite solution (that needs a simple change in get_strings) is to introduce a special comment that get_strings can collect. The false need sometimes to be put in a loop - not very pretty.

Example: /*get_strings: flagname1 */ just near the place you have the flagname1 constant.

Prototype architecture for get_strings.php plugins

My suggestion to this problem is that code that generates strings on the fly should be able to take a parameter "get_strings" or perhapps have a function generate_strings and generate all the strings into the "same" fileapath as the original file with the only difference that its root will be different (e.g. dynamic_strings)

The generation of such files should be initiated by get_strings through a defined interface (e.g. support could be provided to actually generate the files only the data structures / variable names should be defined.

Also, the plugins needs to be registered. My suggestion is that this is done in an directory called get_strings_modules which contains files which include the corresponding .php-file and calls the correct functionality in that file.